(19)

(11) Publication number: 10198726 A

Generated Document.

# PATENT ABSTRACTS OF JAPAN

(21) Application number: 09256929

(51) Intl. Cl.: G06F 17/50 H01L 21/82

(22) Application date: 22.09.97

| | | | |
|---|---|---|---|
| (30) Priority: | 25.09.96 US 96 719610 | (71) Applicant: | VLSI TECHNOL INC |
| (43) Date of application publication: | 31.07.98 | (72) Inventor: | DOCKSER KENNETH A<br>EHMANN GREGORY E |
| (84) Designated contracting states: | | (74) Representative: | |

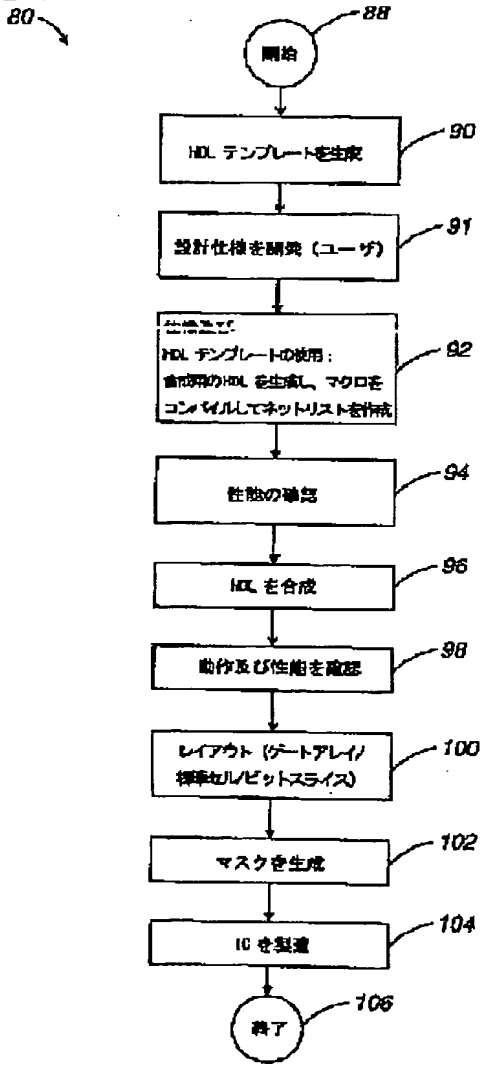
## (54) METHOD AND DEVICE FOR EFFICIENTLY MOUNTING COMPOUND FUNCTION BLOCK IN DESIGN OF INTEGRATED CIRCUIT

(57) Abstract:

PROBLEM TO BE SOLVED: To protect a developer of compound function block design concerning intellectual ownership in a certain degree and to efficiently execute a compound function block.

SOLUTION: A hardware description language(HDL) template is generated (S90), the HDL templates more than one are developed by generating a parameter file and a parameter check file for the HDL template, a design specification to be used for generating the HDL for synthesization is developed (S91), the HDL for synthesization is generated and while using the design specification and the HDL template, a net list for macro blocks more than one is generated (S92).





(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－198726

(43)公開日　平成10年(1998)７月31日

| (51)Int.Cl.[6] | 識別記号 | ＦＩ | |
|---|---|---|---|
| Ｇ０６Ｆ　17/50 | | Ｇ０６Ｆ　15/60 | ６５４Ｋ |
| Ｈ０１Ｌ　21/82 | | Ｈ０１Ｌ　21/82 | Ｃ |

審査請求　未請求　請求項の数20　ＯＬ　（全 21 頁）

(21)出願番号　特願平9－256929

(22)出願日　平成９年(1997)９月22日

(31)優先権主張番号　０８／７１９６１０

(32)優先日　1996年９月25日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　595173488
ヴィエルエスアイ　テクノロジー　インコーポレイテッド
アメリカ合衆国　カリフォルニア州　95131　サン　ホセ　マッケイ　ドライブ　1109

(72)発明者　ケネス　エー.　ダックサー
アメリカ合衆国,　カリフォルニア州,　サン　ノゼ,　テラ　カタ　コート　3132

(72)発明者　グレゴリー　イー.　エマン
アメリカ合衆国,　イリノイ州,　スリーピー　ホロウ,　スーレイ　レーン　825

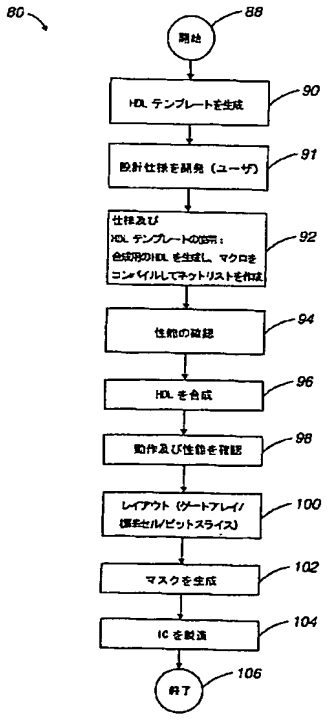
(74)代理人　弁理士　長谷川　芳樹　(外５名)

(54)【発明の名称】　集積回路の設計において複合機能ブロックを効率的に実装する方法及び装置

(57)【要約】

【課題】　複合機能ブロック設計の開発者に対してある程度の知的所有権に関する保護を与えると共に、複合機能ブロックを効率的に実施すること。

【解決手段】　本発明に係る方法は、ＨＤＬテンプレートを生成し、ＨＤＬテンプレートのためのパラメタファイル及びパラメタチェックファイルを生成することにより、１つ以上のＨＤＬテンプレートを開発し、合成用ＨＤＬの生成に際して用いるための設計仕様を開発し、合成用ＨＤＬを生成し、設計仕様、及びＨＤＬテンプレートを使用して１つ以上のマクロブロックのためのネットリストを生成する。



【特許請求の範囲】

【請求項１】　コンパイルツールを用いて集積回路設計を開発するための方法であって、
（Ａ）（ａ）ハードウェア記述言語テンプレートを生成するとともに、（ｂ）前記ハードウェア記述言語テンプレートのためのパラメタファイル及びパラメタチェックファイルを生成することにより、１個以上の前記ハードウェア記述言語テンプレートを開発するステップと、
（Ｂ）合成用ハードウェア記述言語の生成に際して用いるための設計仕様を開発するステップと、
（Ｃ）前記合成用ハードウェア記述言語を生成するステップと、
（Ｄ）前記設計仕様及び前記ハードウェア記述言語テンプレートを使用して１個以上のマクロブロックのためのネットリストを生成するステップと、
を備える方法。

【請求項２】　前記ネットリストを合成するステップと、
合成された前記ネットリストを用いてレイアウトを生成するステップと、
を更に備える請求項１記載の方法。

【請求項３】　合成用に生成された前記ハードウェア記述言語は、前記ハードウェア記述言語で定義されていない１個以上のマクロブロックをインスタンス化する、請求項１記載の方法。

【請求項４】　前記マクロブロックのための前記ネットリストを生成する前記ステップは、前記設計仕様に基づいて前記マクロブロックを目標ライブラリに直接コンパイルするステップを含んでいる、請求項１記載の方法。

【請求項５】　前記合成用ハードウェア記述言語は、前記設計仕様に関連付けられたパスワードのユーザ入力に基づいて前記設計仕様から生成される、請求項１記載の方法。

【請求項６】　前記設計仕様を開発する前記ステップは、前記パラメタファイルに含まれる情報を使用して、ユーザからの入力を受け取るように構成されているメニューを生成するステップ、及び前記メニューを表示するステップを含んでいる、請求項１記載の方法。

【請求項７】　前記設計仕様を開発する前記ステップは、前記パラメタチェックファイルに含まれる情報を使用して前記パラメタファイルに含まれるパラメタの範囲を検査するステップを更に含んでいる、請求項６記載の方法。

【請求項８】　前記合成用ハードウェア記述言語を生成する前記ステップは、
前記ハードウェア記述言語テンプレートを暗号化するステップと、
この暗号化済ハードウェア記述言語テンプレートを解読するステップと、
この解読済ハードウェア記述言語テンプレートを実行するステップと、
を含んでいる、請求項１記載の方法。

【請求項９】　前記解読済ハードウェア記述言語テンプレートによって呼び出された１個以上の従属暗号化済ハードウェア記述言語テンプレートを解読するステップと、
前記解読済従属ハードウェア記述言語テンプレートを実行するステップと、
を更に備えている請求項８記載の方法。

【請求項１０】　前記集積回路設計に用いる１個以上のマスクを生成するステップを更に備える請求項１記載の方法。

【請求項１１】　請求項１０に記載の方法によって形成される集積回路。

【請求項１２】　コンパイルツールを用いて集積回路を設計する方法であって、
（Ａ）（ａ）異なる機能を提供するために複数のハードウェア記述言語テンプレートを生成するとともに、
（ｂ）前記複数のハードウェア記述言語テンプレートのための複数のパラメタファイル及び複数のパラメタチェックファイルを生成することにより、前記複数のハードウェア記述言語テンプレートを開発するステップと、
（Ｂ）合成用ハードウェア記述言語の生成に際して用いるための設計仕様を開発するステップと、
（Ｃ）前記合成用ハードウェア記述言語を生成するステップと、
（Ｄ）前記設計仕様及び前記複数のハードウェア記述言語テンプレートを使用して、１個以上のマクロブロックのためのネットリストを生成するステップと、
を備える方法。

【請求項１３】　前記複数のハードウェア記述言語テンプレートは、１個以上の下位テンプレート、及び、その下位テンプレートを呼び出すように構成されている１つ以上の上位テンプレートを備えている、請求項１２記載の方法。

【請求項１４】　前記複数のパラメタファイルは、前記上位テンプレートに関連する第１のパラメタ、及び前記下位テンプレートに関連する第２のパラメタを備えている、請求項１３記載の方法。

【請求項１５】　前記合成用ハードウェア記述言語を生成する前記ステップは、
前記上位テンプレートを暗号化するステップと、
前記下位テンプレートを暗号化するステップと、
前記上位テンプレートを解読するステップと、
前記下位テンプレートを解読するステップと、
前記下位テンプレートを実行するように構成されている前記上位テンプレートを実行するステップと、
を更に含んでいる、請求項１３記載の方法。

【請求項１６】　中央演算装置、及びこの中央演算装置に接続されている記憶装置を有するコンピュータシステ

ムと、
集積回路設計用の設計仕様を開発する際に用いるための、１個以上のハードウェア記述言語テンプレート、１個以上のパラメタファイル、及び１個以上のパラメタチェックファイルを取得する手段と、
前記設計仕様を開発する手段と、
前記設計仕様及び前記ハードウェア記述言語テンプレートを使用して合成用ハードウェア記述言語を生成する手段と、
前記設計仕様及び前記ハードウェア記述言語テンプレートを使用してネットリストを生成する手段と、
を備える集積回路設計ツール。

【請求項１７】　前記設計仕様開発手段は、前記パラメタファイルに含まれる情報を使用して、ユーザから入力を受け取るように構成されているメニューを生成し、このメニューを表示する手段を含んでいる、請求項１６記載の集積回路設計ツール。

【請求項１８】　前記設計仕様開発手段は、前記パラメタチェックファイルに含まれる情報を使用して前記パラメタファイルに含まれるパラメタの範囲を検査する手段を更に含んでいる、請求項１７記載の集積回路設計ツール。

【請求項１９】　前記設計仕様開発手段は、前記設計仕様を満足させるために用いる適当なマクロを決定する手段を更に含んでいる、請求項１８記載の集積回路設計ツール。

【請求項２０】　前記マクロ決定手段は、前記パラメタファイルを用いて所望のオプションセットを複数生成する手段を含んでいる、請求項１９記載の集積回路設計ツール。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、集積回路の設計に関し、特に、コンパイル（compiltation）及び合成（synthesis）による集積回路の自動化設計に関する。

【０００２】

【従来の技術】過去数十年にわたり、集積回路（ＩＣ）は、近代電子装置に必要不可欠なものとなってきた。「既製」のコンポーネントを使用するよりも、カスタムＩＣまたはセミカスタムＩＣを使用することがしばしば望ましい。簡単な論理設計のカスタムＩＣまたはセミカスタムＩＣにより、特定の性能制約または回路要求を満たし、回路の製造を支援できるように、設計用及び製造用の自動化システムが開発されている。自動化設計システムはまた、ライフサイクルの短いＩＣや非常に複雑なＩＣの設計に広く用いられている。

【０００３】計算機援用設計（ＣＡＤ）システムの一種として、「論理合成システム」が知られている。このようなシステムでは、入力、出力及びハイレベル設計記述がハードウェア記述言語（ＨＤＬ）を用いてコンピュータに入力される。この後、合成ソフトウェアは、ＨＤＬで記述された機能を実行する論理設計を生成する。

【０００４】合成を使用する一般的な集積回路設計および製造プロセスは、通常、レジスタトランスファレベル（ＲＴＬ）のハイレベル設計記述を用いて開始される。ＲＴＬ記述は、例えば、市販のＶＨＤＬまたはVerilog-ＨＤＬといったＨＤＬを用いて生成される。このＨＤＬを合成されることで、ＨＤＬを用いて記述された機能を実行するコンポーネント及びこれらのコンポーネントの相互接続を特定する「ネットリスト」（すなわち、「ネット」のリスト）を生成することができる。しかしながら、現実のウェハ上におけるコンポーネントの配置設計、及び配線接続のトポグラフィ（topography）は、続く「レイアウト」段階のために残される。

【０００５】従来の集積回路の製造方法１０の概略が、図１中にステップ１２から始まるフローチャートによって記載されている。ステップ１４では、一組の設計仕様が開発される。一般的に、これらの仕様には、集積回路全体の性能、及び特定の機能、寸法、及びチップ上におけるコンポーネントの配置特徴が含まれていても良い。

【０００６】論理設計者は、ＨＤＬを用いてステップ１６にて設計される論理のＲＴＬ記述を生成する。共通ハードウェア記述言語には、前述のＶＨＤＬ及びVerilog-ＨＤＬが含まれるが、他の好適な言語（例えば、開発者が知的所有権を保持するＨＤＬ）も使用することができる。この後、設計のＲＴＬ記述は、ステップ１８にてネットリストを合成するために用いられる。ネットリストは、ハードウェア記述言語（ＨＤＬ）によっても記述され得る。ステップ２０は、ネットリストの動作（behavior）及び機能（functionality）を確認し、動作及び機能が仕様を満足しない場合には、再度、ステップ１６及びステップ１８を実行させる。

【０００７】前述のように、ネットリストは、接続されるコンポーネント（「セル」あるいは「モジュール」として知られている）を特定するが、正確な配線形状（wiringtopography）は特定しない。セルは、他のセルのピンと相互接続するためのピンを１本以上有している。「ネットリスト」は、セルのピン間の接続性を定義する「ネット」を含んでいる。換言すれば、「ネット」は、多数のセルの電気的に等価なピンのセットであり、これらのピンは、共通電気ノードを形成するために相互に接続されなければならない。ネットリストにより記述されているコンポーネント、あるいは、セルは、設計仕様を満たす論理設計を形成する。

【０００８】さらに、図１の従来のプロセスを参照すると、ステップ２４にて、論理設計者は、確認済のネットリスト記述をレイアウトツール中に転送する。レイアウトツールにより実行されるレイアウトステップ２４は、集積回路ダイまたはチップの「レイアウトエリア」上におけるセルの実際の物理的配置を決定して、ゲートアレ

イ、あるいは、標準セルを形成する。「レイアウトエリア」は、ＩＣの能動部品（active component）のために割り当てられる領域である。レイアウトプロセスの「配置」ステップは、数十万のゲートを有するＩＣ用コンピュータワークステーション上で計算に数日を要する、非常に時間のかかるステップである。セルのピン間の接続または「ワイヤ」の実際の経路も、レイアウトステップ２４にて決定される。

【０００９】ステップ２４で生成されたカルテック中間フォーマット（C.I.F.）データは、ステップ２６において、集積回路マスクが生成されるマスク製造ツールに転送される。このマスクは、集積回路チップ、または、ダイを生成するために用いられる。このマスクは、C.I.F.データを読み取るために備えられているマシン上で生成される。このC.I.F.データは、ハードディスク、磁気テープ、フロッピーディスク、あるいは、他の伝達媒体を通じてこのマシンに転送することができる。マスク生成マシンは、ネットリストを合成するマシンの一部、あるいは、同一のマシンであっても良い。

【００１０】ステップ２８では、集積回路が製造される。従来の回路製造方法は、フォトリソグラフィプロセスにおいて、ステップ２６で生成されたマスクを使用する。チップ自体が製造されてしまうと、ダイ上の集積回路は、外部回路に対する接続を備えなければならない。これは、一般的に、集積回路にボンディングワイヤ及び／またはリードフレームを取り付けることによって達成される。この後、回路は、プラスティック等のパッケージ材料に封入される。集積回路の設計及び製造は、３０で示されているこの時点で完了する。

【００１１】集積回路がより複雑になるにつれて、論理設計者には、複合機能ブロックを集積回路設計中に迅速に実装する能力を備えていることが要求される。集積回路設計中に複合機能ブロックを迅速に実装する１つの手法としては、固定ネットリストの論理回路へのインスタンス化（instantiation）が挙げられる。論理設計への固定ネットリストのインスタンス化は、マイクロプロセッサ等の固定定義装置用のネットリストに対しては上手く機能するが、様々な構成を備え得るマルチピラー等のブロック用の固定ネットリストのインスタンス化は非効率的である。例えば、設計が１４ビットマルチピラーブロックを要求するが、２４ビットの固定ネットリストマルチピラーしか利用できない場合、得られる設計は、結局、要求されるものよりも大きく、遅いものとなる。

【００１２】集積回路設計中に複合機能ブロックを迅速に実装する他の手法としては、マクロテンプレートを合成ツールに対して供給する前に、設計者によるマクロテンプレート中のパラメタの変更を可能にするＲＴＬで生成されているマクロテンプレートを使用することが挙げられる。マクロテンプレートの合成は、目標ライブラリと詳細には関連付けられないので、この手法により得られる結果は、複合機能ブロックの最適な実装とは言えない。これは、従来の論理合成ツールは、機能を原始コンポーネントまで分解した後、それらを目標ライブラリにバックアップしようと試み、しばしば特定の最適化セルを失うためである。さらに、マクロテンプレートは一般的にＨＤＬコードの羅列なので、マクロテンプレートが顧客に配布される際、マクロテンプレートの開発者は、マクロテンプレートに関連する大量の知的所有権を開示することになる。

【００１３】

【発明が解決しようとする課題】本発明は、上記した従来技術の問題点を解決するためになされたものであり、複合機能ブロック設計の開発者に対してある程度の知的所有権保護を与えると共に、複合機能ブロックを効率的に実装することを目的とする。

【００１４】

【課題を解決するための手段】上記目的を達成するため、請求項１に記載の発明に係るコンパイルツールを用いた集積回路設計の開発方法は、（Ａ）（ａ）ハードウェア記述言語テンプレートを生成するとともに、（ｂ）ハードウェア記述言語テンプレートのためのパラメタファイル及びパラメタチェックファイルを生成することにより、１個以上のハードウェア記述言語テンプレートを開発するステップと、（Ｂ）合成用ハードウェア記述言語の生成に際して用いるための設計仕様を開発するステップと、（Ｃ）合成用ハードウェア記述言語を生成するステップと、（Ｄ）設計仕様及びハードウェア記述言語テンプレートを使用して１個以上のマクロブロックのためのネットリストを生成するステップと、を備えている。

【００１５】また、請求項１２に記載の発明に係るコンパイルツールを用いた集積回路設計の開発方法は、

（Ａ）（ａ）異なる機能を提供するために複数のハードウェア記述言語テンプレートを生成するとともに、

（ｂ）複数のハードウェア記述言語テンプレートのための複数のパラメタファイル及び複数のパラメタチェックファイルを生成することにより、複数のハードウェア記述言語テンプレートを開発するステップと、（Ｂ）合成用ハードウェア記述言語の生成に際して用いるための設計仕様を開発するステップと、（Ｃ）合成用ハードウェア記述言語を生成するステップと、（Ｄ）設計仕様及び複数のハードウェア記述言語テンプレートを使用して、１個以上のマクロブロックのためのネットリストを生成するステップと、を備えている。

【００１６】さらに、請求項１６に記載の発明に係る集積回路設計ツールは、中央演算装置及びこの中央演算装置に接続されている記憶装置を有するコンピュータシステムと、集積回路設計用の設計仕様を開発する際に用いるための、１個以上のハードウェア記述言語テンプレート、１個以上のパラメタファイル、及び１個以上のパラ

メタチェックファイルを取得する手段と、設計仕様を開発する手段と、設計仕様及びハードウェア記述言語テンプレートを使用して合成用ハードウェア記述言語を生成する手段と、設計仕様及びハードウェア記述言語テンプレートを使用してネットリストを生成する手段と、を備えている。

【００１７】本発明は、テンプレート設計者の知的所有権に対してある程度の保護を与えると同時に、カスタム設計を作成するために設計に関するパラメタを論理設計者が修正できるようにする暗号化済ハードウェア記述言語（ＨＤＬ）テンプレートを論理設計者に提供する改良方法を提供する。本発明の主たる利点は、暗号化済ＨＤＬテンプレート及び関連マクロを論理設計者からの入力と有効に結びつけてカスタム出力を提供することができることである。カスタム出力は全てのパラメタを含んではいないので、論理設計者には、特定の実装のためのＨＤＬだけが提供され、ＨＤＬテンプレートに関連する知的所有権の全てが論理設計者に対して明かされることはない。これにより、テンプレート設計者に対して知的所有権に対するある程度の保護を与えると同時に、論理設計者は、複合マクロブロックを効率的に実装することができるようになる。

【００１８】本発明の他の利点は、所定の設計に用いるための最適なマクロが、論理設計者によって提供される入力を用いてＨＤＬテンプレート中にインスタンス化された各マクロをコンパイルすることにより生成されることにある。このことは、設計仕様の達成に役立つばかりでなく、論理設計の品質の向上にも役立つ。さらに、従来の合成ツールとは異なり、マクロブロックコンパイラは、目標ライブラリの詳細な情報を用いて機能する。このコンパイラは、適当な複合セルを小さな領域に直接インスタンス化し、得られるチップの経路決定性（routability）を向上させる。

【００１９】

【発明の実施の形態】以下、本発明に係る集積回路設計中に複合機能ブロックを効率的に実装するための方法および装置を具体化した発明の実施形態について図面を参照して説明する。

【００２０】図２は、本発明に係る集積回路の製造プロセスを示している。このプロセス８０は、ステップ９０で開始する。ステップ９０では、ハードウェア記述言語（ＨＤＬ）のテンプレートが生成される。ＨＤＬテンプレートは、ユーザあるいは論理設計者が「制御」あるいは修正しうる、設定可能なＨＤＬコードの一部である。ＨＤＬテンプレートの生成方法は、図４を参照して後述する。ＨＤＬテンプレートが生成された後、ステップ９１では、ユーザによって提供された情報を用いて設計の仕様が開発される。一般的に、設計仕様の開発ステップは、図１を参照して上述したように、集積回路全体の能力、また、チップ上における特定の寸法、及び特定のコンポーネントの配置に関連するパラメタを含んでいても良い。ユーザにより提供された情報を使用する設計仕様の開発は、図５を参照して後述する。

【００２１】ステップ９２では、ステップ９１にて開発された設計仕様をステップ９０にて生成されたＨＤＬテンプレートと共に用いて、集積回路用ネットリストを合成するために用いることの可能なマクロネットリスト及びＨＤＬが生成される。パラメタ及びパラメタチェックファイルもまた、合成可能なＨＤＬの生成の際に用いることができる。合成可能なＨＤＬ、すなわち「合成用ＨＤＬ」、の生成プロセスに含まれる詳細なステップは、図８を参照して後述する。この後、ステップ９４にて、マクロネットリストの性能が確認される。コンパイル済マクロの性能が仕様を満足しない場合には、新規のコンパイル済マクロを生成しても良い。すなわち、ステップ９２は、好適なマクロネットリストが取得されるまで繰り返すことができる。

【００２２】ＨＤＬの生成を可能にするために、特定のパラメタの組み合わせに固有のパスワードを要求しても良い。このようなパスワードは、特定のホストコンピュータに関連付けられていても良く、また、有効期限を有していても良い。パスワードを使用することにより、ベンダーは、特定の構成についてだけ合成可能ＨＤＬを生成するようユーザを規制することができる。したがって、ユーザは、全ての順列をコンパイルすることによってテンプレートの内容を取得することが不可能となる。

【００２３】マクロネットリストが仕様を満足する場合、ステップ９６にて、合成用ＨＤＬからネットリストが合成される。このネットリストは、ＨＤＬで記述することができ、任意の好適な合成ソフトウェアを使用することによって合成することができる。ステップ９８は、合成されたネットリストの動作の確認である。合成済みのネットリスト及びマクロネットリストの動作または性能が仕様を満足しない場合には、ステップ９８は、ステップ９２，９４及び９６を繰り返し実行させる。合成ネットリストの動作及び性能が確認されると、ステップ１００にて論理設計者は、確認済ネットリストの記述をレイアウトツール中に転送する。レイアウトツールによって実行されるレイアウトステップ１００は、ゲートアレイ、標準セル、あるいは、ビットスライスを形成するために、集積回路ダイ又はチップの「レイアウトエリア」上におけるセルの実際の物理的配置を決定する。セルのピンの間の接続、あるいは、「ワイヤ」の現実の経路もまた、レイアウトステップ１００にて決定される。

【００２４】ステップ１００にて生成されたデータは、一般的に、カルテック中間フォーマット（C.I.F.）であるが、このデータは、ステップ１０２にて、集積回路のマスクが製造されるマスク製造ツールへ転送される。このマスクは、集積回路チップ、あるいは、ダイを生成するために用いられる。マスクの生成は、図１を参照して

説明済みである。

【００２５】マスクが生成された後、ステップ１０４にて集積回路が製造される。従来の回路製造方法は、フォトリソグラフィプロセスにおいて、ステップ１０２で生成されたマスクを使用する。チップ自体が製造されてしまうと、ダイ上の集積回路は、外部回路に対する接続を備えなければならない。これは、一般的に、集積回路にボンディングワイヤ及び／またはリードフレームを取り付けることによって達成される。この後、この回路は、プラスティック等のパッケージ材料に封入される。集積回路の設計及び製造は、１０６で示されているこの時点で完了する。

【００２６】図３には、集積回路製造システム３１のブロック図が図示されている。システム３１は、中央演算装置（ＣＰＵ）３２、Ｉ／Ｏスロット３４、キーボード３６、モニタ３８、ＲＯＭ４０、ＲＡＭ４２、ディスクドライブ装置４４、マスク生成装置４６及びＩＣ製造装置４８を備えている。ＣＰＵ３２はＩ／Ｏポート３４、及び、ユーザ入力装置、例えば、キーボード３６と接続されている。ＨＤＬはＩ／Ｏポート３４、ユーザ入力装置３６、あるいは、他の入力チャネル、例えば、ディスクドライブ４４、を介して受け取ることができ、システム３１に入力することができる。

【００２７】一般的に、Ｉ／Ｏポート３４を介して受け取られるＨＤＬ、あるいは、ＨＤＬテンプレートは、他のマシン（コンピュータ）から送られる。これは、例えば、ネットリストが他のコンピュータによって合成されるときの事例である。ユーザ入力装置３６は、通常、キーボードの形式を採る。これにより、ユーザ、おそらく、論理設計者は、設計仕様を入力し、あるいは、ＣＰＵによって実行されるネットリストマルチピラーを制御することができる。一般的に、入力装置、または、キーボード３６を使用する論理設計者は、ＣＰＵ３２に接続されているモニタ３８を使用する。

【００２８】システムプロセッサ３１には、種々の型式のディジタル記憶容量が設けられていることが好ましい。図３に示されるように、一般的に、このディジタル記憶装置には、ＲＯＭ４０、ＲＡＭ４２、及びディスクドライブ４４が含まれる。ディスクドライブ４４は、Ｉ／Ｏポート３４又はユーザ入力装置３６から受け取ったＨＤＬを記憶し、あるいは、ＨＤＬをシステムに入力するために用いることができ、また、ハードウェア記述プロセッサ３１及びそのＣＰＵ３２上で実行中のプロセスによって生成されるマスク生成データを記憶することができる。ディスクドライブ４４は、他の永久記憶装置、例えば磁気テープやフロッピーディスク、によって置換あるいは増補することができる。既述のように、元のネットリストは、例えば、Ｉ／Ｏポート３４又はユーザ入力装置３６のいずれか一方を通じて入力し、あるいは、システム３１上で直接合成することができる。

【００２９】システム３１は、ネットリストからマスク生成データを開発する。このマスク生成データは、例えば、ディスクドライブ４４のようなディジタル記憶装置に格納される。マスク生成装置４６は、マスク生成データをＣＰＵ３２から受け取る。この他に、（図示しないが）マスク生成装置４６は、マスク生成データを、ディスクドライブ４４等のディジタル記憶装置から直接受け取っても良い。マスク生成装置４６は、ハードウェア記述プロセッサの一部であっても良いし、別の装置であっても良い。このマスク生成データ、あるいは、C.I.F.は、フォトリソグラフィマスクを生成するためにマスク生成装置４６によって用いられる。これらのマスクは、集積回路のコンポーネントをウェハ上に形成するために、集積回路製造装置４８によって使用される。このマスクは、集積回路上のコンポーネントやこれらのコンポーネント間の接続を形成するために十分なものである。集積回路製造装置４８は、当業者に周知の半導体製造設備、例えばエッチング装置、化学的気相成長（ＣＶＤ）機、リソグラフィ機等、を有している。

【００３０】システム３１によるプロセスの最終生成物は、パッケージされた集積回路５０である。このパッケージＩＣ５０は、マスク生成装置４６によって生成されたマスクを使用して生成されたダイ５２を含んでいる。この半導体ダイ５２は、一般的に、ディジタル集積回路、及び回路をリード５３に接続するためのＩ／Ｏパッド５１を有している。Ｉ／Ｏパッド５１は、任意の従来方法、例えば、ボンディングワイヤによってリード５３と接続することができる。

【００３１】既述のように、ネットリストは、コンポーネント（セル）及びそれらの理論的相互接続を順に特定するネットを記述する。換言すれば、ネットリストは、論理設計の理論的な記述である。複合論理設計に対しては、テンプレートを用いることで、ＲＴＬを合成ツールに入力する前に論理設計者が論理設計に関連するパラメタを修正できるようにすることができる。

【００３２】図４は、ＨＤＬテンプレート生成プロセスを詳細に示す。このプロセス９０は、３００にて始まり、ステップ３０２では、種々の機能に固有の多数のＨＤＬテンプレートが生成される。一つの実施形態では、これらの種々の機能はハイレベル機能であっても良い。ステップ３０４では、パラメタファイル及びパラメタチェックファイルが各ＨＤＬテンプレートについて生成される。各ＨＤＬテンプレートのパラメタに関連する部分は、ＨＤＬテンプレートから取得され、パラメタファイルを生成するために用いられる。このため、パラメタファイルは、パラメタファイルに関連付けられたＨＤＬテンプレートに対するグローバルパラメタを定義するために用いられる情報を含んでいる。

【００３３】図５を参照して後述するように、このパラメタファイルは、通常、ＨＤＬテンプレート用のユーザ

入力ウィンドを定義するために用いることのできる情報も含んでいる。すなわち、このパラメタファイルは、通常、グラフィカルユーザインターフェイス（ＧＵＩ）を定義するために用いられる。一般的に、コンパイルツールに固有のスクリプト言語で記述されるパラメタチェックファイルは、パラメタファイル内のパラメタを検査してパラメタが正当なエントリであるか否かを判定するために用いられる。換言すれば、パラメタチェックファイルは、パラメタファイル上にて誤り検査を行うために用いられる。パラメタファイル及びパラメタチェックファイルの例は、図１０、並びに図１１及び図１２にそれぞれ示されている。

【００３４】ステップ３０６では、ＨＤＬテンプレートが暗号化される。テンプレートを暗号化するために用いることのできるサードパーティ製暗号化及び解読ソフトウェアツールは、「FLEXcrypt」の製造元であるカリフォルニア州キャンベル市のグローブトロッタソフトウェア（Globetrotter Software, Inc.）を含む多くのベンダーから入手可能である。ＨＤＬテンプレート及びマクロの生成及び暗号化のプロセスは、３０８にて完了する。

【００３５】図５は、設計仕様開発プロセスを詳細に示す。このプロセス９１は４００にて始まり、ステップ４０２では、パラメタファイルを用いてメニューが生成され、スクリーンディスプレイ上に表示される。パラメタファイル内のメニュー関連情報には、例えば、ディスプレイ中のメニュー項目のタイトルの定義に用いることの可能な「メニュータイトル」が含まれていても良い。表示されるメニューの内容は、図６を参照して後述する。メニューが生成されて表示された後、ステップ４０４にてユーザからの入力を受け取る。この入力には、パラメタ形式の設計仕様や、パラメタを使用するオペレーションの実行要求や、設計仕様開発プロセスの中断要求が含まれる。

【００３６】ステップ４０６では、ユーザから受け取った入力のタイプに関する判定がなされる。この実施形態では、パラメタ、ユーザ入力受け取りプロセスの中断要求、あるいは、メニュー生成に用いられたパラメタファイルに関連するパラメタのコンパイルまたは評価、のいずれのタイプに入力が当てはまるか判定される。入力タイプがパラメタであると判定された場合、プロセス制御は、ステップ４０４に戻る。入力タイプがユーザ入力受け取りプロセスの中断要求であると判定された場合、本プロセスは４２２で完了する。

【００３７】ステップ４０６にて、入力はパラメタファイル中のパラメタのコンパイルまたは評価要求であると判定された場合には、プロセス制御は、ステップ４０８に進む。ステップ４０８では、適当なパラメタチェックファイルを用いてパラメタ範囲が検査される。すなわち、このパラメタチェックファイルは、パラメタチェックファイルに関連付けられたパラメタファイル中の入力済パラメタ上で誤り検査を実行する。この誤り検査の後、ステップ４１０では、入力済パラメタのいずれかが範囲を外れていないかどうか判定される。１個以上のパラメタが範囲を外れていると判定された場合には、ステップ４１２にてスクリーン上にエラーメッセージが表示される。エラーメッセージが表示された後、プロセス制御は、ユーザに範囲外のパラメタを変更する機会を与えるために、ユーザからの入力を受け取るステップ４０４に戻る。

【００３８】ステップ４１０にて、範囲外のパラメタが１個もないと判定された場合には、プロセス制御は、ステップ４１４に進む。ステップ４１４では、ステップ４０４にてユーザから受け取った入力が、ＨＤＬテンプレートを評価し、結果として得られる設計に用いられるマクロブロックの適切なバージョンを決定することの要求であったか、あるいは、コンパイル要求であったかが判定される。入力が設計の評価要求であった場合には、ステップ４１６にて、パラメタに関連付けられたＨＤＬテンプレートが解読ツールを用いて解読される。ＨＤＬテンプレートが解読されると、現実の設計において用いるための、最良の、あるいは、最適なマクロブロックアーキテクチャがステップ４１８にて決定される。コンパイラツールは、要求されたマクロブロックの異なるアーキテクチャについてのネットリストを生成し、最良のものを選択する。例えば、マクロブロックが加算器で用いられる場合には、といった、加算器の幾つかの異なるアーキテクチャ、例えばキャリルックアヘッド（carry-look ahead）やキャリスキップ（carry skip）、がコンパイルされ、得られるネットリストのサイズ、タイミング、及び他の属性が解析されることになる。最良のマクロブロックは、例えばタイミングや性能といった点において特定の設計仕様を満足するのに最も役立つマクロブロックである。最適マクロブックのアーキテクチャが識別された後、プロセス制御は、ステップ４０４に戻る。ステップ４１２にて、入力はコンパイル要求、すなわち、コンパイル命令、であると判定された場合には、設計仕様の開発プロセスは、４２０で完了する。

【００３９】図６は、パラメタファイル中に設けられた情報から生成されるメニュー表示を図示する。メニュー５００は、一般的に、ＧＵＩであり、ユーザがキーボード、及び好適なポインティング装置、例えばマウス、を用いてメニュー５００のフィールドにアクセスできるようにモニタ上に表示される。図示の実施形態では、メニュー５００は、同期先入れ先出し（ＦＩＦＯ）メモリに使用される入力情報を取得するように設定される。メニュー５００は、ユーザにより与えられる情報の全タイトル、すなわち「同期ＦＩＦＯテンプレートコンパイラ」を定義するために用いられるメニュータイトル５０２、を有している。

【００４０】ＦＩＦＯと連携するランダムアクセスメモリ（ＲＡＭ）に関する種々の物理パラメタは、ユーザが与えることができる。例えば、「ワード奥行（word depth）」５０４及び「ワード幅（word width）」５０６を特定することが可能である。ワード奥行５０４及びワード幅５０６は共に、それぞれ、ＲＡＭについての奥行及び幅の表示を与える整数である。「ＲＡＭタイプ」５０８は、ＦＩＦＯのメモリコアについての構成を表示する。図示の実施形態では、ユーザは、論理設計の要件に応じて、ＲＡＭタイプ５０８をラッチタイプ５１０、フリップ／フロップタイプ５１２、あるいは、ＲＡ５タイプ５１４の中から選択する。

【００４１】ＦＩＦＯのメモリコアが満杯であるか否かを示すために用いられる「フルフラグ」５１６信号の追加は、ユーザが選択することができる。同様に、メモリコアが空であるか否かを示す「エンプティフラグ」５１８も選択することができる。ＦＩＦＯのうち何ワードがデータを含んでいるかを示す「ＦＩＦＯ奥行インジケータバス」５２０も選択することが可能である。

【００４２】ラッチ５１０がＲＡＭタイプ５０８として選択される場合、論理設計者が望むなら、テストモードピンを有するラッチを具体化、あるいはインスタンス化することができる。「ラッチＲＡＭ用スキャンテストモードピン」選択５２２は、入力選択ボックス５２４から適当なスキャニングオプションを選択することにより実行することができる。入力選択ボックス５２４中のスキャニングオプションには、図示のようにＲＡＭに対して用いられるラッチはスキャン型であるべきでないことを示す「なし」オプションが含まれ得る。他のオプションには、ＲＡＭに対して用いられるラッチはテストモードピンに対するバッファ信号（buffered signal）を含むことを示す「バッファ」オプション、及びあらゆる種類のバッファリングを用いないでラッチの全テストモードピンが互いに接続されるべきであることを示す「非バッファ」オプションが含まれる。

【００４３】「コンポーネント名」５２６は、設計されているコンポーネントの名称を特定するために用いられる。ある実施形態では、デフォルトのコンポーネント名５２８は、メニュー５００によって特定されるコンポーネントの名称であっても良い。例えば、図示するように、デフォルトのコンポーネント名５２８は、「fifo」である。同様に、「出力ファイル名」５３０は、コンポーネント名５２６によって識別されたコンポーネントを含むファイル名を特定するために用いられる。デフォルト出力ファイル名５３２は、コンポーネントの名称と同じにすることができ、図示の実施形態では「fifo」である。

【００４４】メニュー５００はまた、コマンドを選択して呼び出すために活性化されるインターフェースを有している。「評価」コマンド５３４は、図５を参照して前述したように、入力済パラメタの範囲を検査したり、入力済パラメタを用いて生成されたテンプレートを解読したり、テンプレート中に与えられた設計仕様を満足するために用いる最良のマクロブロックアーキテクチャを決定するために選択することができる。「コンパイル」コマンド５３６は、図５を参照して前述したように、入力済パラメタの範囲を検査したり、テンプレートを解読したり、入力済パラメタを使用して合成用ＨＤＬを生成したり、マクロブロックをコンパイルするために選択することができる。「閉じる」コマンド５３８は、メニュー５００を用いた情報入力プロセスを中止または終了させるために選択することができる。ある実施形態では、「ヘルプ」コマンド５４０が選択されると、メニュー５００中のいくつかのパラメタを定義することが可能な情報を含む「ヘルプメニュー」が表示されるようにすることができる。他の実施形態では、ヘルプコマンド５４０は、メニュー５００中に図示されるパラメタに関連するヘルプメニューを含み、特定のヘルプメニューにアクセスするために用いることが可能な情報を含む総合メニューを表示しても良い。

【００４５】図７は、設計仕様を開発する際に用いる適当なマクロブロックアーキテクチャを決定するプロセスを詳細に示している。このプロセス４１８は、６００にて始まり、ステップ６０２では、所望のオプションセットのリストを生成するため、あるいは、所望のパラメタ組み合わせのリストを生成するために、ＨＤＬテンプレートに関連付けられたパラメタファイルが用いられる。ある実施形態では、パラメタファイルをトラバースすることで、全ての可能性あるパラメタ組み合わせの順列（permutation）が見つかる。これらの順列は、この後、所望のオプションセットのリストを生成するために用いられる。他の実施形態では、所望のオプションセットのリストは、全ての可能性あるオプションセットのサブセットである。ステップ６０４では、オプションセットのリストから１個のオプションセットが選択される。ステップ６０６では、未だ選択及び解析のされていないオプションセットが残っているかどうかが判断される。選択すべきオプションセットが残っていない場合、論理設計の合成に用いるための適当なマクロブロックの決定のために所望のオプションセットの全てが既に選択及び解析された旨が表示され、適切なマクロブロックを選択するプロセスは６１２にて完了する。

【００４６】ステップ６０６において、選択可能な他の所望のオプションセットが残っていると判断された場合、プロセスフローは、ステップ６０８に進む。ステップ６０８では、ＨＤＬテンプレートにインスタンス化されたマクロブロックが選択オプションセットを用いてコンパイルされる。ＨＤＬテンプレートを用いてインスタンス化された各マクロブロックは、選択されたオプションセットと共同して各マクロブロックの性能を判定する

ために呼び出され、コンパイルされる。

【００４７】ステップ６１０では、コンパイル済マクロブロックが解析される。そして、解析の結果は、通常、論理設計者が読むことのできるレポートの中で評価および提示される。このレポートには、オプションセットを用いてコンパイルされた各マクロブロックの論理設計のサイズ、速度、及び間隙率（porosity）、すなわち実際のチップ上に与えられた設計の経路決定の容易さの尺度、の評価が含まれていても良い。但し、これらに限られるわけではない。解析及び続く評価の結果は、通常、所期の目的に用いるために最適なマクロブロックを決定するために用いられる。コンパイル済マクロブロックが解析及び評価されると、プロセス制御は、ステップ６０４に戻り、次のオプションセットを選択する。プロセス制御がステップ６０４に戻ると、マクロブロックを用いてすでにコンパイルされたオプションセットに基いて、所期の目的に用いるために最適なマクロブロックに関する情報が利用可能になる。

【００４８】図８は、設計仕様及びＨＤＬテンプレートを用いて合成用ＨＤＬを生成するプロセスの詳細を示している。合成用ＨＤＬを生成するために用いられるテンプレートが暗号化される。プロセス９２は７００にて始まり、ステップ７０２では、テンプレートファイルへのポインタ及びパラメタをテンプレートコンパイラが受け取る。ポインタが指示するテンプレートファイルには、「上位」テンプレート、及び上位テンプレートによって使用される「下位」テンプレートが含まれる。指示されるテンプレートファイルは、暗号化済テンプレートファイルである。ステップ７０４では、テンプレートファイルは、解読ツール、例えば図４を参照して前述したサードパーティ製の解読ツール、を用いて解読される。一般的に、上位テンプレートファイル、及び上位テンプレートファイルによって要求される下位テンプレートファイルのみが解読される。

【００４９】上位テンプレートファイルは、ステップ７０６にて実行される。この上位テンプレートファイルの実行により、全ての必要なサブファイル、あるいは、上位テンプレートファイルが要求することの可能な従属下位テンプレートファイルが実行される。ある実施形態では、下位テンプレートファイルが、他の下位テンプレートファイルの実行を引き起こしても良い。合成用ＨＤＬ生成プロセスにおける上位テンプレートファイルの実行ステップ、及び必要なサブファイルの実行ステップは、呼出のネスティングに備える。すなわち、各サブファイルは、実行されるべきサブファイルが無くなるまで継続して実行される。サブファイルは、順次に他のサブファイルを呼び出すことができる。当業者であれば理解できるように、ファイル又はサブファイルの実行は、再帰的な実行（recursive execution）という結果になっても良い。すなわち、上位ファイル、あるいは、サブファイルは、一連の呼出を通じて自身を直接的、あるいは、間接的に呼び出すことができる。本実施形態では、この再帰（recursion）は、オペレーティングシステムによって処理される。テンプレートファイルの実行の詳細は、図９を参照して後述する。上位テンプレートファイル、及び全ての必要なサブファイルの実行により、合成用ＨＤＬの生成およびマクロブロックのネットリストへのコンパイルが行われる。この後、合成用ＨＤＬの生成プロセスは７０８にて完了する。

【００５０】図９は、テンプレートファイル実行プロセスを詳細に示している。このプロセス７０６は、８００にて始まる。ステップ８０２では、テンプレートファイルからコマンドが選択される。選択されたコマンドは、一般的に、テンプレートファイル中にて入手可能な第１の非実行コマンドライン、あるいは、次の命令である。上位テンプレートファイルは、コード化されたコマンドをも含む場合があるが、上位テンプレートファイルは、通常、下位テンプレートファイルを呼び出すように機能する。上位ファイルの例は、図１３に見ることができる。ステップ８０４では、テンプレートファイル（これは上位テンプレートファイルであっても良い）からの最後のコマンドが実行されたかどうか判定される。実行されるべき命令が残っていない場合には、テンプレートファイル実行プロセスは、８１０にて完了する。テンプレートファイル実行プロセスは再帰的である。すなわち、最初の上位テンプレートからの全ての命令が実行されるまで、合成用ＨＤＬを生成しインスタンス化マクロをコンパイルするプロセスは継続する。すなわち、プロセス７０６が繰り返し実行されることになる。

【００５１】実行されるべきコマンドが存在する場合には、プロセス制御は、ステップ８０６に進む。ステップ８０６では、どのタイプの命令がテンプレートファイルから選択されたかが判定される。本実施形態では、次の４つのタイプのコマンドを選択することができる。すなわち、マクロブロックプログラムを呼び出すコマンド、他のテンプレートファイルを実行するコマンド、上位テンプレートファイル中に含まれるコード化済コマンドを実行するコマンド、及びテンプレートグラフィカルユーザインターフェース（ＧＵＩ）へ戻るコマンドの四つである。

【００５２】呼び出されるコマンドのタイプがマクロブロックプログラムを呼び出すコマンドである場合、プロセス制御は、ステップ８０６からステップ８０８に移行する。ステップ８０８では、マクロブロックプログラム、あるいは、コンパイラが呼び出される。マクロブロックプログラムは、ＧＵＩにて特定されたパラメタに基いて適当なマクロブロック用のネットリストを作成する。マクロブロックプログラムが呼び出された後、プロセス制御は、ステップ８０２に戻り、実行されるべきテンプレートファイルから次のコマンドが選択される。

【００５３】呼び出されるコマンドのタイプがテンプレートファイル実行の呼出であるとステップ８０６で判定された場合には、図８を参照して前述したように、ステップ８１０にて合成用ＨＤＬ生成ステップ全体が繰り返し実行され、任意の要求暗号化テンプレートファイルが解読される。テンプレートファイルが解読された後、プロセス制御はステップ８０２に戻る。再帰法がサポートされているので、呼び出されているテンプレートファイルは、呼出元テンプレートと同一であっても良い。

【００５４】呼び出されたコマンドのタイプがテンプレートＧＵＩに戻る要求であるとステップ８０６で判定された場合には、プロセス制御はステップ８０６からステップ８１８に進む。ステップ８１８では、解読ツールが呼び出され、未だ解読されていない任意の要求テンプレートファイルが解読される。換言すれば、図８を参照して前述したように、合成用ＨＤＬ生成ステップ全体が繰り返し実行され、任意の要求暗号化テンプレートファイルが解読される。例えば、上位テンプレートファイルはＵＡＲＴに対して固有であるが、ＦＩＦＯメモリに関連する情報が上位テンプレートファイルによって呼び出される場合、ＦＩＦＯメモリに関連するテンプレートファイルを解読する必要がある可能性がある。ある実施形態では、ＧＵＩを用いることで、論理設計者が「新規」な、依然として暗号化されている、解読されるべきテンプレートファイルに関連する設計仕様を入力することができるようになる。他の実施形態では、論理設計者からのいかなる新規入力をも用いることなく、新規のテンプレートファイルに関連するパラメタファイル中の仕様が用いられる。テンプレートファイルが解読された後、プロセス制御はステップ８０２に戻る。

【００５５】テンプレートファイルが解読され、ネットリストを用いて論理設計がレイアウトされた後は、図２を参照して前述したように、集積回路マスクを生成するためにマスク製造ツールを用いることができる。この集積回路マスクは、集積回路チップまたはダイを生成するために用いられる。このマスク生成装置は、ネットリストを合成するために用いられる機械と同じもの又はその一部であっても良い。一般的に、フォトリソグラフィプロセスでは、マスクを用いて集積回路が生成される。チップ自身が製造されると、ダイ上の集積回路は外部回路に対する接続を備えていなければならない。これは、一般的に、集積回路にボンディングワイヤ及び／またはリードフレームを取り付けることによって達成される。この後、回路は、プラスティック等のパッケージ材料に封入され、これにより、集積回路が製造される。

【００５６】以上、いくつかの発明の実施の形態に基づき本発明を説明したが、本発明の趣旨から逸脱しない範囲で種々の変更物、改良物、均等物が存在する。また、本発明に係る方法および装置の双方を実現する他の種々の方法が存在することにも留意すべきである。

【図面の簡単な説明】

【図１】　従来技術に係る集積回路の設計及び製造プロセスを示すフローチャートである。

【図２】　本発明に係る集積回路の設計及び製造プロセスを示すフローチャートである。

【図３】　集積回路の形成に用いられるシステムを示すブロック図である。

【図４】　ＨＤＬテンプレート生成プロセスを詳細に示すフローチャートである。

【図５】　設計仕様開発プロセスを詳細に示すフローチャートである。

【図６】　パラメタファイル中に備えられている情報に基づき生成されるメニュー表示を示す説明図である。

【図７】　最良マクロブロックの決定プロセスを詳細に示すフローチャートである。

【図８】　合成用ＨＤＬを生成するために設計仕様及びＨＤＬテンプレートを使用するプロセスを詳細に示すフローチャートである。

【図９】　テンプレートファイル実行プロセスを詳細に示すフローチャートである。

【図１０】　パラメタファイル及びパラメタチェックファイルの第１の例を示す説明図である。

【図１１】　パラメタファイル及びパラメタチェックファイルの第２の例を示す説明図の前半部分である。

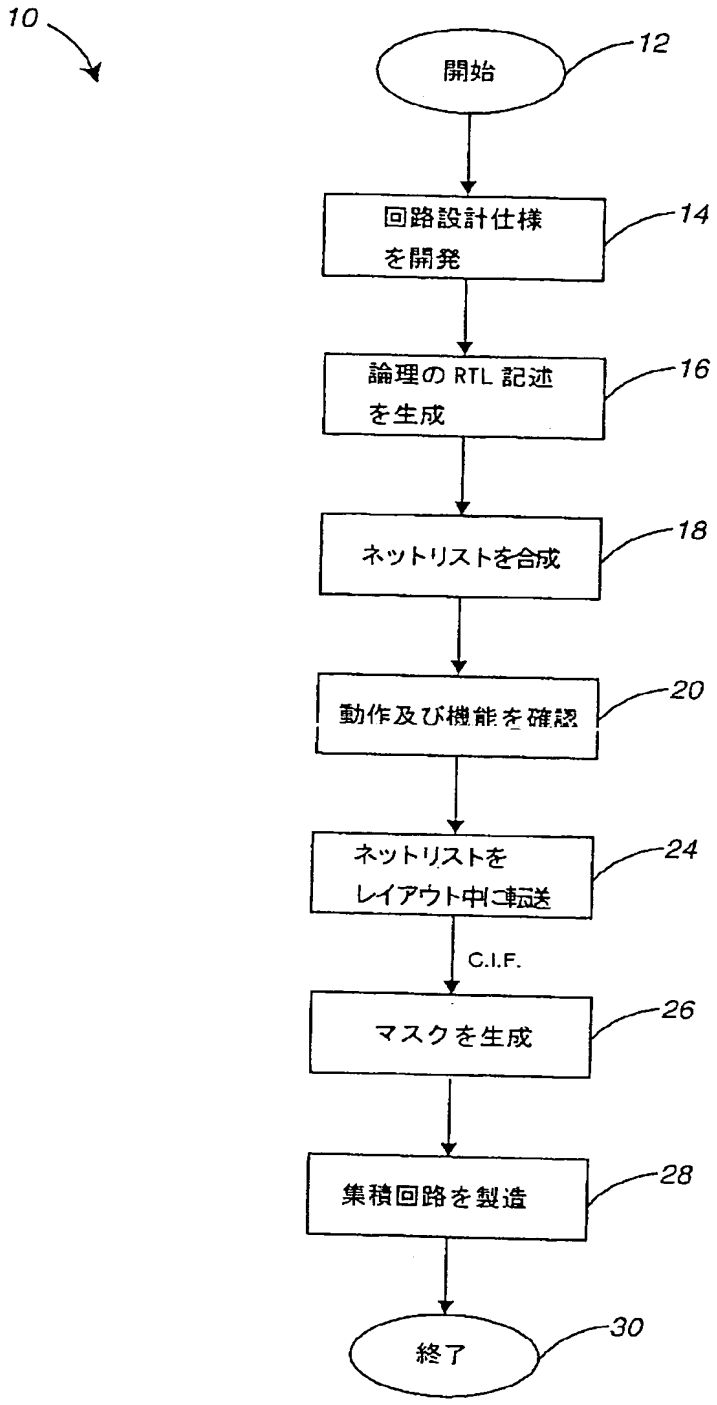
【図１２】　パラメタファイル及びパラメタチェックファイルの第２の例を示す説明図の後半部分である。

【図１３】　上位ファイルを例示する説明図である。

【符号の説明】

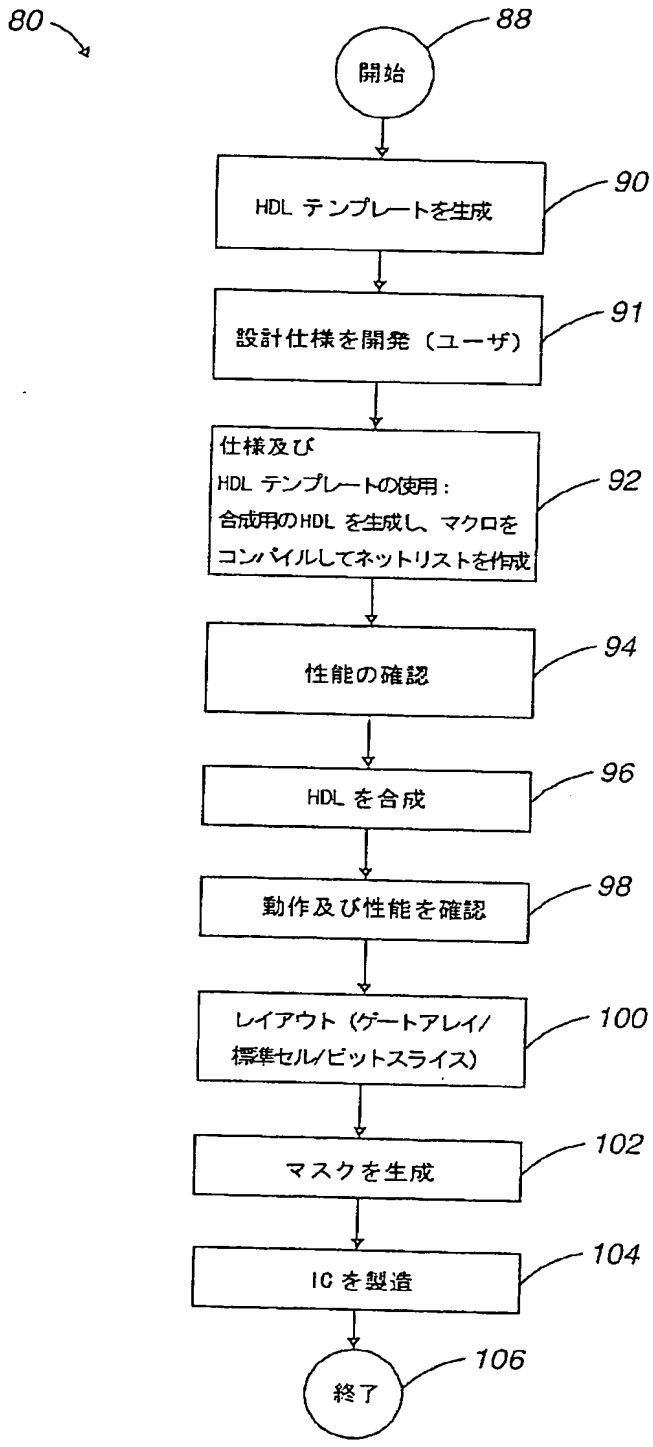
３１…集積回路製造システム、３２…中央演算装置（ＣＰＵ）、３４…Ｉ／Ｏスロット、３６…キーボード、３８…モニタ、４０…ＲＯＭ、４２…ＲＡＭ、４４…ディスクドライブ、４６…マスク生成装置、４８…ＩＣ製造装置、５０…パッケージＩＣ、５２…ダイ、５３…リード、５００…メニュー、５０４…ワード奥行、５０６…ワード幅。

【図１】

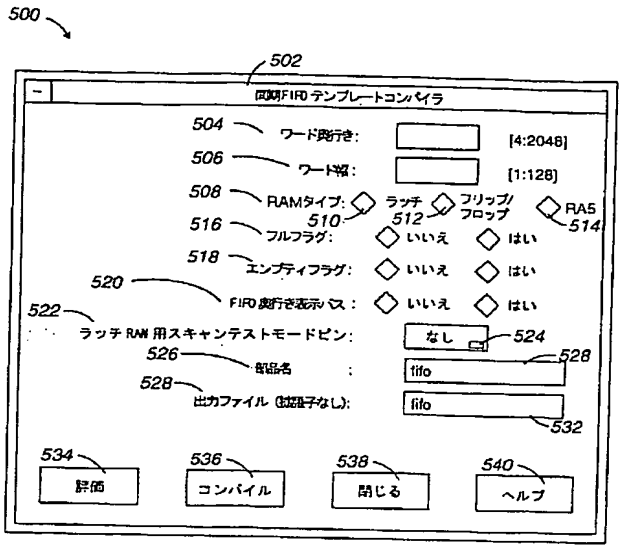

10
開始
12
回路設計仕様
を開発
14
論理の RTL 記述
を生成
16
ネットリストを合成
18
動作及び機能を確認
20
ネットリストを
レイアウト中に転送
24
C.I.F.
マスクを生成
26
集積回路を製造
28
終了
30

従来技術

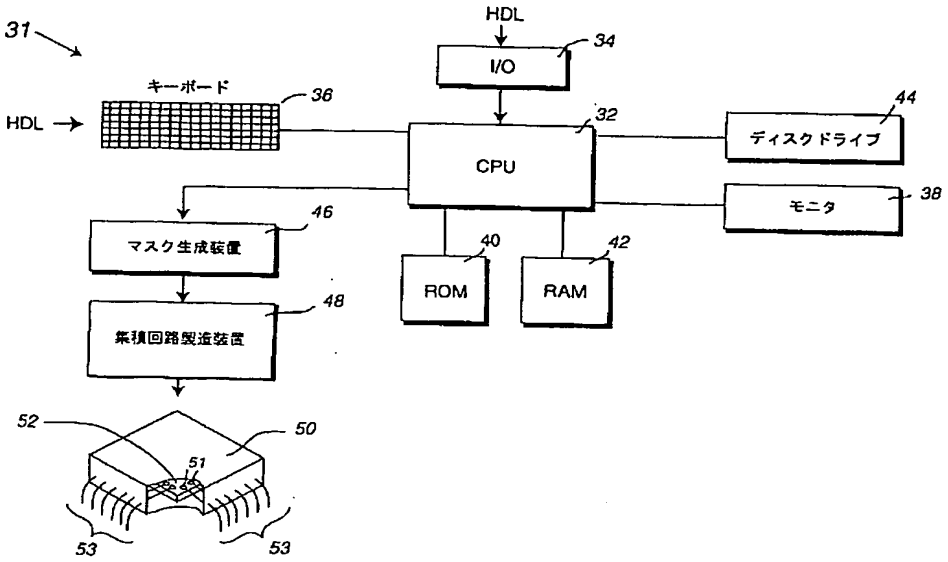
【図２】


80
88
開始
HDL テンプレートを生成
90
設計仕様を開発（ユーザ）
91
仕様及び
HDL テンプレートの使用：
合成用のHDL を生成し、マクロを
コンパイルしてネットリストを作成
92
性能の確認
94
HDL を合成
96
動作及び性能を確認
98
レイアウト（ゲートアレイ/
標準セル/ビットスライス）
100
マスクを生成
102
IC を製造
104
終了
106

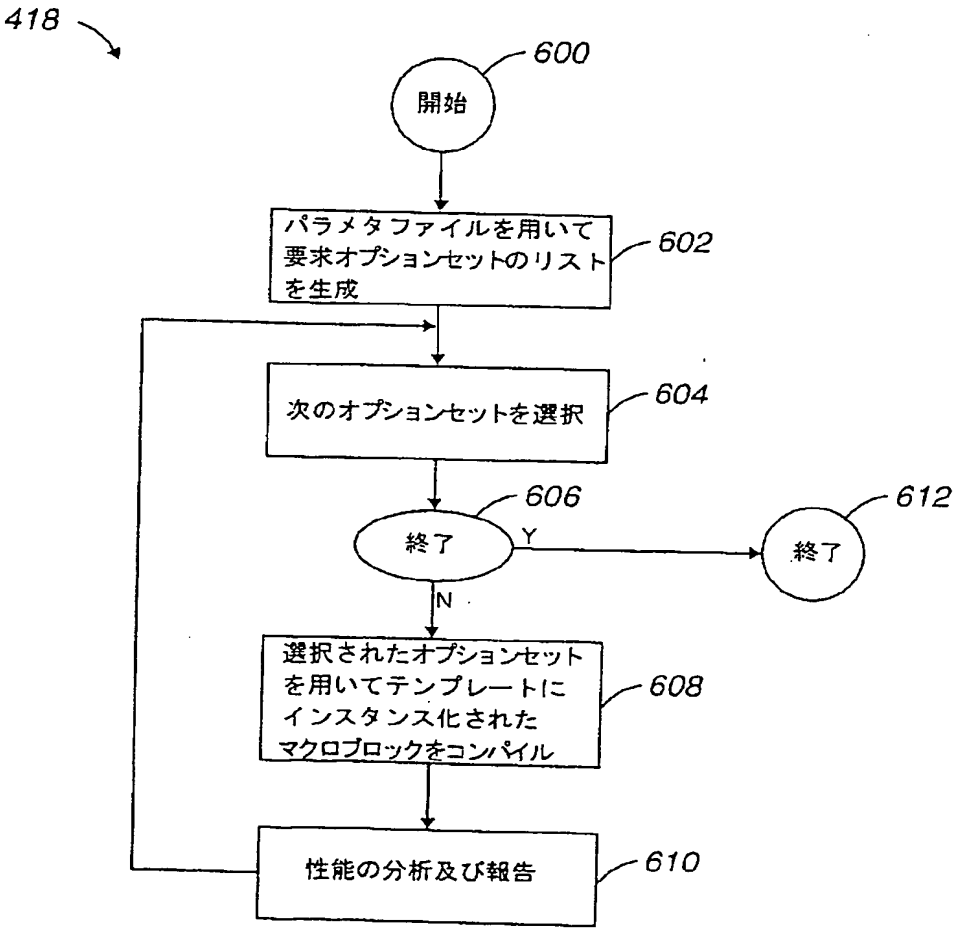
【図６】


500
502
FIFO テンプレートコンパイラ
504
ワード長さ：
[4:2048]
506
ワード幅：
[1:128]
508
RAMタイプ：
ラッチ
フリップ/
フロップ
RA5
510
512
514
516
フルフラグ：
いいえ
はい
518
エンプティフラグ：
いいえ
はい
520
FIFO 実行表示バス：
いいえ
はい
522
ラッチ RAM 用スキャンテストモードピン：
なし
524
526
部品名
：
fifo
528
528
出力ファイル（拡張子なし）：
fifo
532
534
評価
536
コンパイル
538
閉じる
540
ヘルプ

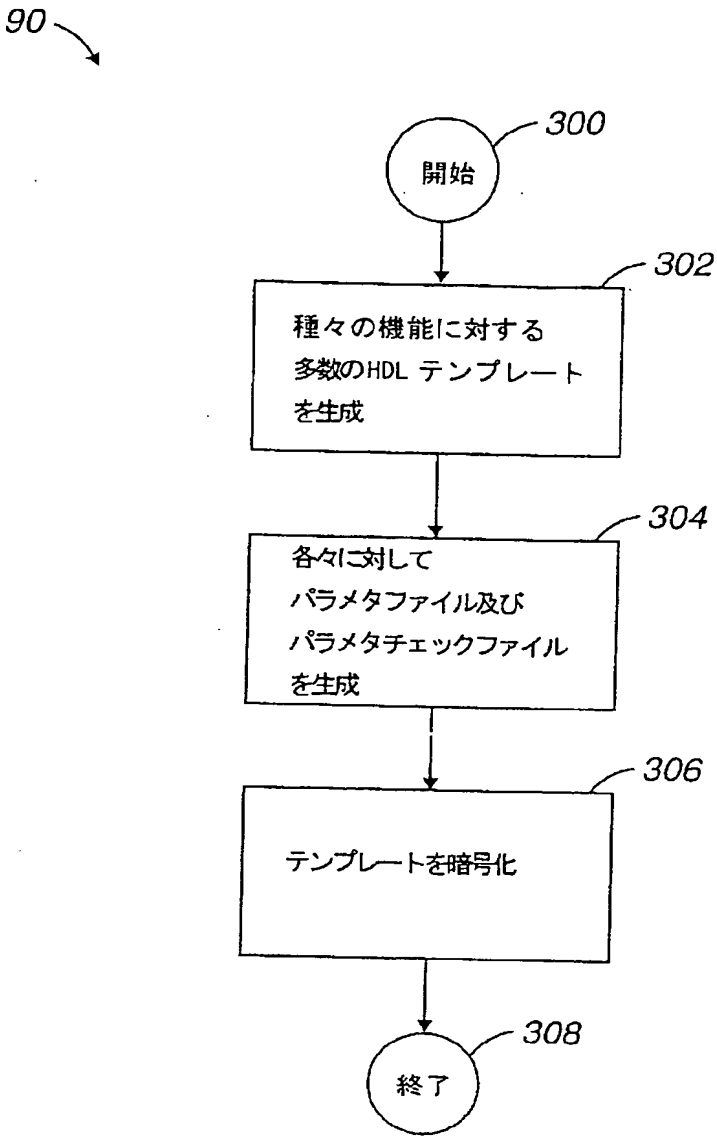
【図３】


31
HDL
I/O
34
キーボード
36
HDL
32
CPU
44
ディスクドライブ
モニタ
38
46
マスク生成装置
40
ROM
42
RAM
48
集積回路製造装置
52
50
51
53
53

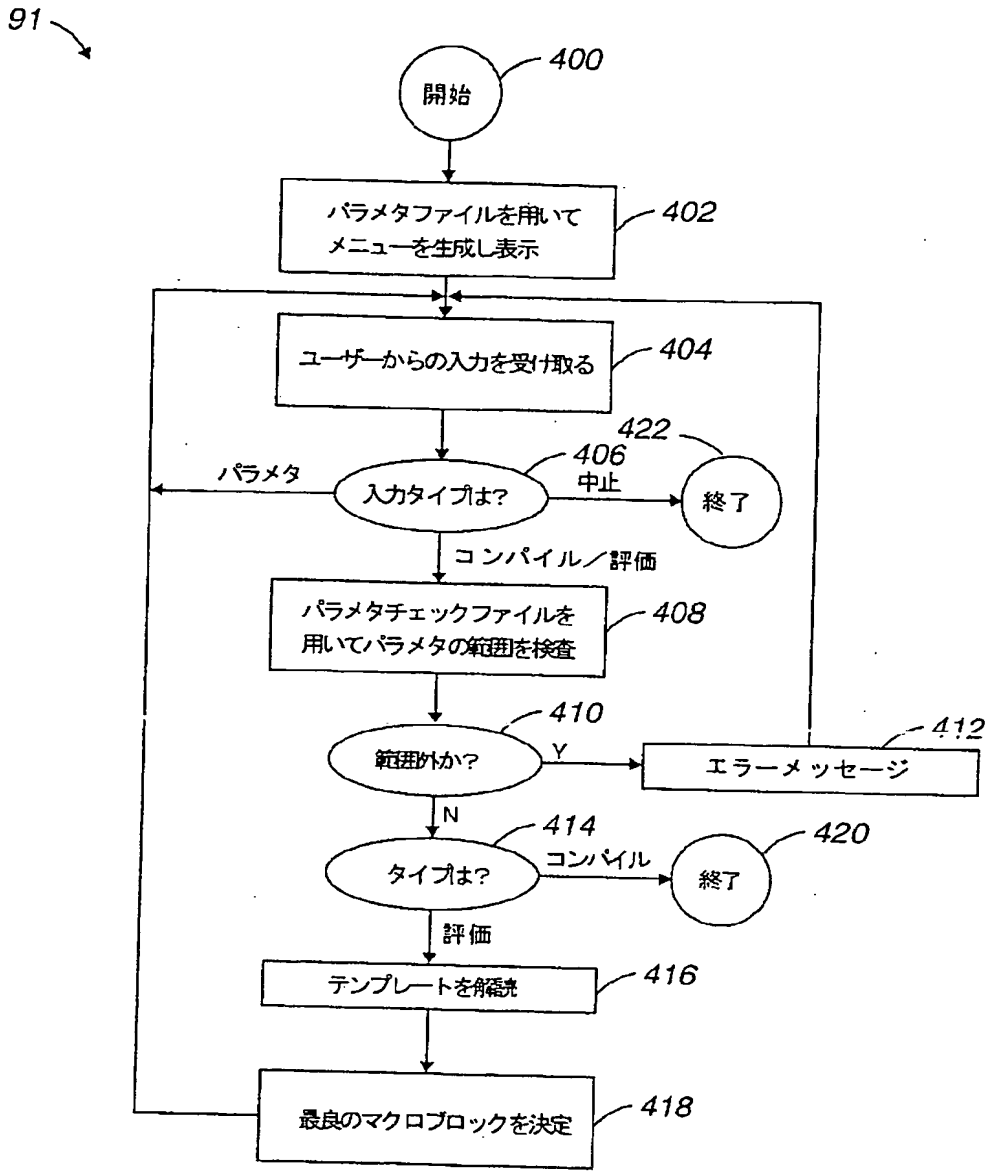
【図７】


418
600
開始
パラメタファイルを用いて
要求オプションセットのリスト
を生成
602
次のオプションセットを選択
604
606
終了
Y
612
終了
N
選択されたオプションセット
を用いてテンプレートに
インスタンス化された
マクロブロックをコンパイル
608
性能の分析及び報告
610

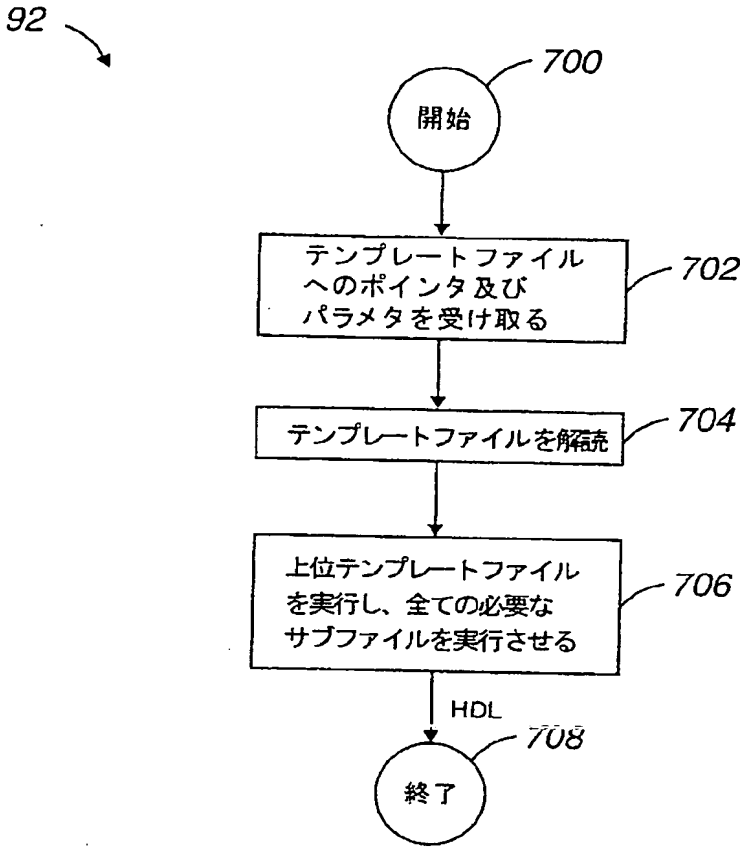
【図４】


90
300
開始
302
種々の機能に対する
多数のHDL テンプレート
を生成
304
各々に対して
パラメタファイル及び
パラメタチェックファイル
を生成
306
テンプレートを暗号化
308
終了

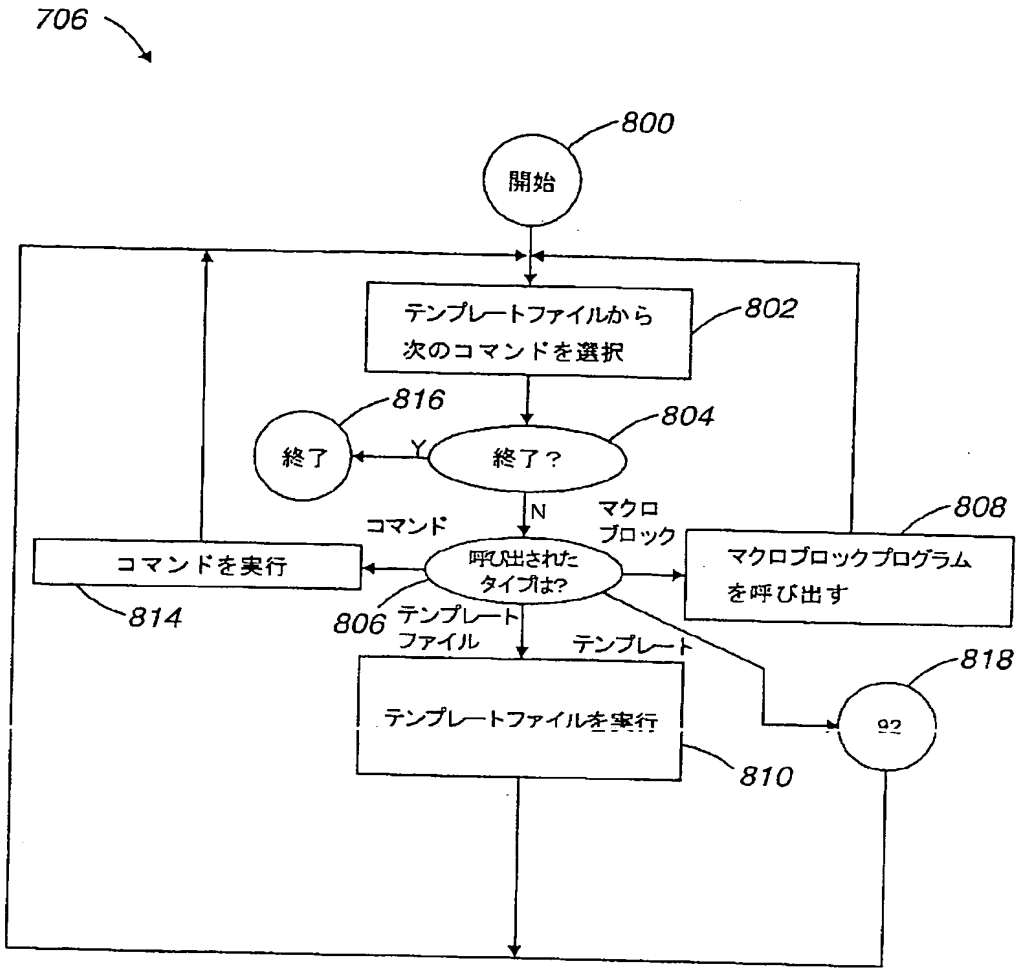
【図５】


91
400
開始
402
パラメタファイルを用いて
メニューを生成し表示
404
ユーザーからの入力を受け取る
422
406
パラメタ
入力タイプは?
中止
終了
コンパイル／評価
408
パラメタチェックファイルを
用いてパラメタの範囲を検査
410
412
範囲外か?
Y
エラーメッセージ
N
414
420
タイプは?
コンパイル
終了
評価
416
テンプレートを解読
418
最良のマクロブロックを決定

【図8】


92
700
開始
テンプレートファイル
へのポインタ及び
パラメタを受け取る
702
テンプレートファイルを解読
704
上位テンプレートファイル
を実行し、全ての必要な
サブファイルを実行させる
706
HDL
708
終了

【図9】


706
800
開始
テンプレートファイルから
次のコマンドを選択
802
816
終了
Y
804
終了？
N
コマンド
マクロ
ブロック
808
コマンドを実行
呼び出された
タイプは?
マクロブロックプログラム
を呼び出す
814
806
テンプレート
ファイル
テンプレート
818
テンプレートファイルを実行
92
810

【図10】

```
TITLE       "Synchronous Interface FIFO - 1 Clock - Template"
MENUTITLE     "Synchronous Interface FIFO - 1 Clock"
CHECKFILE     "fifo.hpc"
VERILOGHDLFILE "fifo_ver.htf"
VHDLHDLFILE   "fifo_vhd.htf"
HELPFILE      "sisfifo.pdf"
SHORTNAME     "sisfifo"
VERSION       "1.0"
TEMPLATEENABLED

DESCRIPTION   -
"This is a fifo template compiler, with synchronous read and write \n-
port interfaces and a single clock for read and write port operations.\n-
Providing Full, Empty, and Depth indications. \n-
\n-
Example:\n-
 To compile a 8x8 fifo with Full & Empty flags\n-
 hdli -shell sisfifo -FE -d 8 -b 8 -o test"


#  variable   GUI                    Shell Field      Right Hand  Elements
#  name       question               tag  type        Text
#=============================================================================
TP | DEPTH    | "Word Depth:"          | -d | INTEGER    | "[4:2048]" | | | "Word
depth"
TP | WIDTH    | "Word Width:"          | -b | INTEGER    | "[1:128]" | | | "Word
width"
TP | RAMTYPE  | "Memory Type:"         | -t | RADIO      |        | | |-
 "Latch" "Flip/Flop" "RA5" |-
 "Use Latches for the memory core." -
 "Use Flip/Flops for the memory core." -
 "Use a compiler RA5 ram for the memory core."
TP | SCAN | "Scan Testmode pin for Latched Memory:" | -T | OPTION  | | | "None"
        "Buffered" "Unbuffered" | "No Testmode provided" "Buffered Testmode pin
        provided" "Unbuffered Testmode pin provided"
TP | DECODE | "Read Decode Type for Latched Memory:" | -D | RADIO  | | 0 |-
        "Muxed" "Tristate" |-
        "Use Muxes to generate the Read decode" "Use Tristates to generate the Read
        decode"
TP | RESET    | "Controller Reset Type:"  | -r | RADIO      | | 0 | "Asynchronous"
        "Synchronous" |-
        "Provide a asynchronous FIFO controller reset" "Provide a synchronous FIFO
        controller reset"
TP | FULL     | "Full Flag?"           | +F | RADIO      |        | | | "No" "Yes" |-
        "Provide full flag"
TP | EMPTY    | "Empty Flag?"          | +E | RADIO      |        | | | "No" "Yes" |-
        "Provide empty flag"
TP | LEVELBUS | "FIFO Depth Indicator Bus?" | +L | RADIO      |        | | "No"
        "Yes" | "Provide Depth bus"
```

【図11】

```
#print("Synchronous Interface - Synchronous FIFO - Version ",VERSION,"\n");

#if (DEPTH < 4)
# error("The \"Word Depth\" must be greater than 4.\n");
# exit(1);
#endif
##
#if (DEPTH > 2048)
# error("The \"Word Depth\" must be less than or equal to 2048.\n");
# exit(1);
#endif
##
#if ( DEPTH*WIDTH > 16384 && RAMTYPE == 2)
# error("The total number of bits for a RA5 RAM must be less than 16384.\n");
# exit(1);
#endif
##
#if ( DEPTH > 64 && RAMTYPE < 2)
# error("If the depth is greater than 64 the \"RAM Type\"\nmust be of a compiled type
(ie. RA5).\n");
# exit(1);
#endif
##
#if (WIDTH < 1)
# error("The \"Word Width\" must be greater than 0.\n");
# exit(1);
#endif
##
#if (WIDTH > 128)
# error("The \"Word Width\" must be less than or equal to 128.\n");
# exit(1);
#endif
##
#if (SCAN && RAMTYPE == 1)
# error("A Flip/Flop memory is fully SCAN testable, so this options is not
needed.\n");
# exit(1);
#endif
##
#if (SCAN && RAMTYPE > 1)
# error("The Scan option is only available for Latch based memories.\n");
# exit(1);
#endif
##
```

【図１２】

```
#if (RAMTYPE == 2 && (substr(TARGETLIB,1,4) != "vsc7" &&
substr(TARGETLIB,1,4) != "vsc6"))
# error("The RA5 Ram is only supported in cell-based .6um and .5um
technologies.\n");
# exit(1);
#endif
##
#if (RAMTYPE == 2 && DEPTH < 8)
# error("The RA5 Ram only supports depths greater than 8.\n");
# exit(1);
#endif
##
#if (ESTONLY)
All options are valid for this template. Push the "View DOC" button for
more information.
#endif
#
#include fifo_conns.htf
#if (LANGUAGE & LANG_VERILOG)
#include fifo_ver_doc.htf
#endif
#
#if (LANGUAGE & LANG_VHDL)
#include fifo_vhd_doc.htf
#endif
#
#exit(0) ;
```

【図13】

```
#string RAM_NAME;
#string DLY_NAME;
#int status;
#int statusall;
#status = 0;
#statusall = 0;
#RAM_NAME = FILENAME "_ram";
#DLY_NAME = FILENAME "_dly";
#
## BUILD THE PORT LIST
#include fifo_ver_head.htf
#
#if (RAMTYPE == 0)
## LATCH RAM
#include fifo_ver_0.htf
#endif
#
#if (RAMTYPE == 1)
## FF RAM
#include fifo_ver_1.htf
#endif
#
#if (RAMTYPE == 2)
## RA5 RAM
#include fifo_ver_2.htf
#endif
#
## BUILD THE TESTBENCH
#include fifo_ver_tb.htf
#
#if (TOOLSET == "Synopsys")
## BUILD THE SYNOPSYS SYNTHESIS SCRIPT
#include fifo_ver_syn.htf
#endif
#
#if (TOOLSET == "Compass")
## BUILD THE COMPASS SYNTHESIS SCRIPT
#include fifo_ver_cmp.htf
#endif
#
#exit(statusall);
```